EXPRESS MAIL NO. EL 746 760 130 US

DATE OF DEPOSIT 8/8/01

Our File No. 9281-4144
Client Reference No. J US98135

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Takayuki Sugawara

Serial No. To Be Assigned

Filing Date: Herewith

For: Arithmetic Operation Unit Suitable for Correcting Lost Data by General-Purpose Computer

SUBMISSION OF CERTIFIED COPY OF PRIORITY DOCUMENT

Commissioner for Patents
Washington, D.C. 20231

Dear Sir:

Transmitted herewith is a certified copy of priority document Japanese Patent Application No. 2000-243774, filed August 11, 2000 for the above-named U.S. application.

Respectfully submitted,

Gustavo Siller, Jr.
Registration No. 32,305
Attorney for Applicant

BRINKS HOFER GILSON & LIONE
P.O. BOX 10395
CHICAGO, ILLINOIS 60610
(312) 321-4200

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年　８月１１日

出願番号
Application Number: 特願２０００－２４３７７４

出願人
Applicant(s): アルプス電気株式会社

２００１年　６月　４日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０５１９１９

【書類名】　特許願
【整理番号】　981212AL
【提出日】　平成12年 8月11日
【あて先】　特許庁長官殿
【国際特許分類】　G11B 20/18
【発明の名称】　演算処理装置
【請求項の数】　5
【発明者】
　【住所又は居所】　東京都大田区雪谷大塚町１番７号　アルプス電気株式会社内
　【氏名】　菅原　孝幸
【特許出願人】
　【識別番号】　000010098
　【氏名又は名称】　アルプス電気株式会社
　【代表者】　片岡　政隆
【代理人】
　【識別番号】　100085453
　【弁理士】
　【氏名又は名称】　野▲崎▼　照夫
【手数料の表示】
　【予納台帳番号】　041070
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　演算処理装置

【特許請求の範囲】

【請求項１】　数１で表されるガロア体上の原始多項式Ｇ（ｘ）のＧ（ｘ）＝０の元をαとしたときに、α^{i}を乗算する演算処理装置であって、

【数１】

$$G_{(x)} = g_m x^m + g_{m-1} x^{m-2} + g_{m-2} x^{m-2} + \cdots + g_{p+1} x^{p+1} + g_p x^p + \cdots + g_0$$

乗算前の元をｉビットシフトするシフト演算部と、αの乗数をｉとしたときに2^{i}個の要素のルックアップテーブルを参照する参照部とを有することを特徴とする演算処理装置。

【請求項２】　前記数１で表されるガロア体上の原始多項式Ｇ（ｘ）のＧ（ｘ）＝０の元をαとしたときに、以下の数２で示されるガロア体上の元Ｕを基に$U \cdot \alpha^{i}$を演算する演算処理装置であって、

【数２】

$$U = \alpha^{n} u_n + \alpha^{n-1} u_{n-1} + \cdots + \alpha^{2} u_2 + \alpha^{1} u_1 + u_0$$

前記Ｕの元をｉビットシフトするシフト演算部と、Ｕの最下位のｉビットに応じて、2^{i}個の要素のルックアップテーブルを参照する参照部との排他的論理和をとることを特徴する演算処理装置。

【請求項３】　データをD_1，D_2，…，D_kとしたときに以下の数３で示されるエラー検査シンボルE_0，E_1，E_2，…，E_{n-k-1}が演算される請求項２記載の演算処理装置。

【数３】

$$
\begin{aligned}
&D_1+D_2+D_3+\cdots+D_{k-1}+D_k = E_0\\
&\alpha^{k}D_1+\alpha^{k-1}D_2+\alpha^{k-2}D_3+\cdots+\alpha^{2}D_{k-1}+\alpha D_k = E_1\\
&\alpha^{(\alpha)^2}D_1+\alpha^{(\alpha-1)^2}D_2+\alpha^{(\alpha-2)^2}D_3+\cdots+\alpha^{4}D_{k-1}+\alpha^{2}D_k = E_2\\
&\qquad\vdots\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\
&\alpha^{(\alpha)^{n-k-1}}D_1+\alpha^{(\alpha-1)^{n-k-1}}D_2+\cdots+\alpha^{n-k}D_{k-1}+\alpha^{n-k-1}D_k = E_{n-k-1}
\end{aligned}
$$

【請求項４】　データをデコードしたときに、以下の数４の演算を行ってシンボルS_0，S_1，S_2，…，S_{n-k-1}を得る請求項３記載の演算処理装置。

【数４】

$$
\begin{aligned}
&D_1+D_2+D_3+\cdots+D_{k-1}+D_k+E_0 = S_0\\
&\alpha^{k}D_1+\alpha^{k-1}D_2+\alpha^{k-2}D_3+\cdots+\alpha^{2}D_{k-1}+\alpha D_k+E_1 = S_1\\
&\alpha^{(\alpha)^2}D_1+\alpha^{(\alpha-1)^2}D_2+\alpha^{(\alpha-2)^2}D_3+\cdots+\alpha^{4}D_{k-1}+\alpha^{2}D_k+E_2 = S_3\\
&\qquad\vdots\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\
&\alpha^{(\alpha)^{n-k-1}}D_1+\alpha^{(\alpha-1)^{n-k-1}}D_2+\cdots+\alpha^{n-k}D_{k-1}+\alpha^{n-k-1}D_k+E_{n-k-1} = S_{n-k-1}
\end{aligned}
$$

【請求項５】　前記シンボルS_0，S_1，S_2，…，S_{n-k-1}を用いてエラーの大きさを求める際、次の逆元参照テーブルを設けて、これを参照して前記エラーの大きさを求める請求項４記載の演算処理装置。

ａ）α^{1}、α^{2}、…、α^{k}、

ｂ）$1+\alpha^{1}$、$1+\alpha^{2}$、…、$1+\alpha^{k}$

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ガロア体上の元を高速に乗算できる演算処理装置に係り、特にデータの消失訂正を汎用の計算機を行うような場合に適した演算処理装置に関する。

【０００２】

【従来の技術】

記録媒体へのデータの記録および再生の際などに行われるエラー訂正処理では、検査シンボルE_0，E_1，E_2，…が付加コードとして記録媒体にエンコードさ

れ、再生時にはデータ（ユーザシンボル）に前記検査シンボルE_0，E_1，E_2，…の排他的論理和をとったシンボルS_0，S_1，S_2，…が求められる。そしてこのシンボルを演算することにより、エラーの大きさが算出される。

【０００３】

前記シンボルE_0とS_0は、単純な排他的論理和により計算が可能であるが、E_1，S_1以降では、αのべき乗算の計算を行う必要がある。ここで、ガロア体上の原始多項式Ｇ（ｘ）が０となるときの元がαである。

【０００４】

記録再生装置などでは、ガロア体の専用演算部を持てないために、ルックアップテーブルを用いて前記検査シンボルＥやデコード時のシンボルＳを求めている。

【０００５】

【発明が解決しようとする課題】

従来は、前記ルックアップテーブルとして８ビット（２５６）×８ビット（２５６）＝６４ｋバイトのテーブルを用いている。よって８ビットずつのデータに対しては、検査シンボルＥやデコード時のシンボルＳを求めることが可能である。しかし、現在では計算機のアクセス幅が１６ビットや３２ビット単位となっているため、１６ビット単位や３２ビット単位でシンボルの演算を行うことが好ましい。

【０００６】

しかし、１６ビット単位の演算では、１６ビット（６４ｋバイト）×１６ビット（６４ｋバイト）＝４Ｇバイトとなり、前記のようなルックアップテーブルを構築することは実質的に不可能である。

【０００７】

またべき乗表現の変換テーブルを使用することも考えられるが、この場合も１６ビット単位で数百ｋバイトオーダのテーブルが必要になる。

【０００８】

本発明は上記従来の課題を解決するものであり、ユーザシンボルが何ビットであっても、ガロア体の元αを用いたべき乗計算を高速に行え、例えば消失訂正を

高速に実現できるようにした演算処理装置を提供することを目的としている。

【０００９】

【課題を解決するための手段】

本発明は、前記数１で表されるガロア体上の原始多項式Ｇ（ｘ）のＧ（ｘ）＝０の元をαとしたときに、α^{i}を乗算する演算処理装置であって、

乗算前の元をｉビットシフトするシフト演算部と、αの乗数をｉとしたときに2^{i}個の要素のルックアップテーブルを参照する参照部とを有することを特徴とするものである。

【００１０】

また本発明は、前記数１で表されるガロア体上の原始多項式Ｇ（ｘ）のＧ（ｘ）＝０の元をαとしたときに、前記数２で示されるガロア体上の元Ｕを基に$U \cdot \alpha^{i}$を演算する演算処理装置であって、

前記Ｕの元をｉビットシフトするシフト演算部と、Ｕの最下位のｉビットに応じて、前記ルックアップテーブル参照結果との排他的論理和をとることを特徴するものである。

【００１１】

また、データ（ユーザシンボル）をD_1，D_2，…，D_kとしたときに前記数３で示されるエラー検査シンボルE_0，E_1，E_2，…，E_{n-k-1}が演算される。

【００１２】

および／または、データ（ユーザシンボル）をデコードしたときに、以下の数４の演算を行ってシンボルS_0，S_1，S_2，…，S_{n-k-1}を得ることができる。

【００１３】

また、前記シンボルS_0，S_1，S_2，…，S_{n-k-1}を用いてエラーの大きさを求める際、次の逆元参照テーブルを設けて、これを参照して前記エラーの大きさを求めることが可能である。

【００１４】

ａ）α^{1}、α^{2}、…、α^{k}、

ｂ）$1+\alpha^{1}$、$1+\alpha^{2}$、…、$1+\alpha^{k}$

【００１５】

【発明の実施の形態】

本発明は、計算機のアクセス幅（例えば１６ビットや３２ビットなど）に一致する次元のガロア体上の原始多項式Ｇ（ｘ）＝０の元をαとしたときに、高速なα^{i}の乗算を行えるようにするものである。本発明の演算処理装置では、計算機のアクセス幅が大きくても高速な乗算が可能であり、従来のように８ビット単位でしかエラー訂正ができなかったという問題を解消することができる。

【００１６】

以下の表１は、磁気ディスクなどに記録されるデータの１ブロック単位のフォーマットを示している。

【００１７】

【表１】

【表1】

Packet No	Double Word 0	Double Word 1	Double Word 2		Double Word 127
0	Byte 0−3 (D_1)	Byte 4−7	Byte 8−11	.....	Byte 508−511
1	Byte 0−3 (D_2)	Byte 4−7	Byte 8−11	.....	Byte 508−511
.....	.....	.....	.....	.....	.....
63	Byte 0−3 (D_{63})	Byte 4−7	Byte 8−11	.....	Byte 508−511
64	ECC[0]	ECC[0]	ECC[0]	.....	ECC[0]
65	ECC[1]	ECC[1]	ECC[1]	.....	ECC[1]

【００１８】

上記表１の縦方向はPacket Noであり、０～６３までが、ユーザブロックであり、ここにはデータ（ユーザシンボル）が含まれる。Packet Noの６４と６５がエラー訂正のための検査シンボルE_0とE_1が含まれる。この検査シンボルE_0とE_1がエラー訂正のためのコードである。

【００１９】

Packet Noの０から６３までのデータは３２ビット（４バイト）のDouble Word単位で、Packet Noの０から６３にかけて演算が施され、各Double Word単位で検査シンボルE_0が演算されて、Packet No６４内に記録される。また各Double Word単位で検査シンボルE_1が演算されて、Packet No６５内に記録される。

【００２０】

記録媒体に表１で示すフォーマットでデータを記録する際に、各Double Word単位で検査シンボルE_0が演算される。この演算の一般式を以下の数５に示す。以下の数５において、D_1, D_2, D_3, …, D_kは、各Double Word単位でのPacket Noの０から６３までの例えばそれぞれ３２ビットのユーザシンボルである。なお表１の場合、前記シンボル数ｋは６３である。

【００２１】

E_0, E_1, E_2, …, E_{n-k-1}は検査シンボルであり、検査シンボル数はｎ－ｋである。なお、表１は検査シンボル数が２の場合を示している。

【００２２】

【数５】

$$
\begin{aligned}
&D_1+D_2+D_3+\cdots+D_{k-1}+D_k = E_0\\
&\alpha^{k}D_1+\alpha^{k-1}D_2+\alpha^{k-2}D_3+\cdots+\alpha^{2}D_{k-1}+\alpha D_k = E_1\\
&\alpha^{(\alpha)^2}D_1+\alpha^{(k-1)^2}D_2+\alpha^{(k-2)^2}D_3+\cdots+\alpha^{4}D_{k-1}+\alpha^{2}D_k = E_2\\
&\qquad\vdots\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\
&\alpha^{(\alpha)^{n-k-1}}D_1+\alpha^{(k-1)^{n-k-1}}D_2+\cdots+\alpha^{n-k}D_{k-1}+\alpha^{n-k-1}D_k = E_{n-k-1}
\end{aligned}
$$

【００２３】

次に、記録媒体からデータをデコードするときには、以下の数６の演算が行われる。この演算によるシンボルS_0, S_1, S_2, …, S_{n-k-1}により後に示すように、データのエラーの大きさが求められる。

【００２４】

【数６】

$$
\begin{aligned}
&D_1+D_2+D_3+\cdots+D_{k-1}+D_k+E_0=S_0\\
&\alpha^{k}D_1+\alpha^{k-1}D_2+\alpha^{k-2}D_3+\cdots+\alpha^{2}D_{k-1}+\alpha D_k+E_1=S_1\\
&\alpha^{(\alpha)^2}D_1+\alpha^{(\alpha-1)^2}D_2+\alpha^{(\alpha-2)^2}D_3+\cdots+\alpha^{4}D_{k-1}+\alpha^{2}D_k+E_2=S_3\\
&\qquad\vdots\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\
&\alpha^{(\alpha)^{n-k-1}}D_1+\alpha^{(\alpha-1)^{n-k-1}}D_2+\cdots+\alpha^{n-k}D_{k-1}+\alpha^{n-k-1}D_k+E_{n-k-1}=S_{n-k-1}
\end{aligned}
$$

【００２５】

エンコードとデコード時での前記数５と数６は、シンボル数マイナス１までのα^{i}乗算器があれば演算可能である。なおｉは０，１，２，３，…，ｋである。なお、前記数１以下の各式における「＋」は排他的論理和（ＥｘＯＲ）である。

【００２６】

まず、前記数５と数６における検査シンボルE_0とS_0は排他的論理和により単純に計算可能であるが、E_1とS_1以降の計算は、べき乗算が必要である。図１はこの計算を行う演算回路の一例を示す。図１に示すように、レジスタ内にD_1，D_2，D_3，…，D_kを格納しておき、それぞれにα，α^2，…，α^{n-k-1}をべき乗算していくことにより、E_1，E_2，…，E_{n-k-1}およびS_1，S_2，…，S_{n-k-1}の演算が可能である。

【００２７】

そこでこのべき乗算を高速化する本発明の計算方法および演算処理装置について説明する。

【００２８】

ガロア体ＧＦ（２）の原始多項式を以下の数７とする。

【００２９】

【数７】

$$G(x)=g_m x^m+g_{m-1}x^{m-2}+g_{m-2}x^{m-2}+\cdots+g_{p+1}x^{p+1}+g_p x^p+\cdots+g_0$$

【００３０】

この演算処理装置の原理を簡単に説明できるように、ここではガロア体の原始

多項式を以下の数８を例として説明する。

【００３１】

【数８】

$$G_{(x)} = X^8 + X^4 + X^3 + X^2 + 1$$

【００３２】

また、数４と数５において、ユーザシンボルD_1，D_2，D_3，…，D_kに対応するシンボルを８ビットのu_7，u_6，u_5，u_4，u_3，u_2，u_1，u_0とすると、U（u_7，u_6，u_5，u_4，u_3，u_2，u_1，u_0）は、以下の数９となる。

【００３３】

【数９】

$$U = \alpha^7 u_7 + \alpha^6 u_6 + \alpha^5 u_5 + \alpha^4 u_4 + \alpha^3 u_3 + \alpha^2 u_2 + \alpha^1 u_1 + u_0$$

【００３４】

またα・Uは以下の数１０のようになる。

【００３５】

【数１０】

$$\alpha \cdot U = \alpha^8 u_7 + \alpha^7 u_6 + \alpha^6 u_5 + \alpha^5 u_4 + \alpha^4 u_3 + \alpha^3 u_2 + \alpha^2 u_1 + \alpha u_0$$

【００３６】

上記数１０に示されるα・Uは、数９のUの元においてu_7，u_6，u_5，u_4，u_3，u_2，u_1，u_0を１ビットシフトさせて、以下の数１１の通りとし、これに数８の原理多項式から得られた数１２を加算（排他的論理和）したものに等しい。

【００３７】

【数１１】

$$\alpha^{7}u_{6}+\alpha^{6}u_{7}+\alpha^{5}u_{4}+\alpha^{4}u_{3}+\alpha^{2}u_{1}+\alpha^{1}u_{0}$$

【００３８】

【数１２】

$$\alpha^{8}\ (=\alpha^{4}+\alpha^{3}+\alpha^{2}+1)\cdot u_{7}$$

【００３９】

よってＵからα・Ｕへの演算では、Ｕの最下位ビットｕ７が「１」の場合と「０」の場合において、２通りのルックアップテーブル（参照部，表２）を、シフト演算部と共に設けることで、前記演算を高速に行うことができる。

【００４０】

【表２】

【表2】

u7	ルックアップテーブルの値
0	0
1	$\alpha^{4}+\alpha^{3}+\alpha^{2}+\alpha^{1}$

【００４１】

次に、α^{2}・Ｕは以下の数１３の通りである。

【００４２】

【数１３】

$$\alpha^2 \cdot U = \alpha^9 u_7 + \alpha^8 u_6 + \alpha^7 u_5 + \alpha^6 u_4 + \alpha^5 u_3 + \alpha^4 u_2 + \alpha^3 u_1 + \alpha^2 u_0$$

【００４３】

これは、前記数９のＵの元を２ビットシフトさせ、以下の数１４を足した（排他的論理和）ものに等しい。

【００４４】

すなわち、Ｕの元の下位２ビット値に応じた2^2＝４通りのルックアップテーブル（表３）を用意しておき、Ｕの２ビットシフト演算との排他的論理和をとれば良い。

【００４５】

【表３】

【表3】

u6, u7	ルックアップテーブルの値
０ ０	0
０ １	$\alpha^4+\alpha^3+\alpha^2+\alpha^1$
１ ０	$\alpha(\alpha^4+\alpha^3+\alpha^2+\alpha^1)$
１ １	$\alpha(\alpha^4+\alpha^3+\alpha^2+\alpha^1)$ $+\alpha^4+\alpha^3+\alpha^2+\alpha^1$

【００４６】

【数１４】

$$\alpha^9(=\alpha\cdot(\alpha^4+\alpha^3+\alpha^2+1))\cdot u_7 + \alpha^8(=\alpha^4+\alpha^3+\alpha^2+1)\cdot u_6$$

【００４７】

シフト演算は、ＣＰＯに一般的に備わっており、またルックアップテーブルを小容量のメモリで実現できるから、汎用かつ高速な演算方法である。

【００４８】

以上は$\alpha^3 \cdot U$，$\alpha^4 \cdot U$においても同じである。すなわちα^iのためには、

１）ｉビットのシフト演算部

２）2^iの数のルックアップテーブル（参照部）

を設けることにより高速演算が可能である。

【００４９】

次に、前記シンボルS_0，S_1，S_2，…，S_{n-k-1}によるエラー訂正について説明する。

【００５０】

前記記録媒体から再生したデータにエラーが無い場合には、数６においてシンボルS_0，S_1，S_2，…，S_{n-k-1}は全て０になる。またユーザシンボル中にエラーが生じた場合、エラーが生じている箇所が予め分かっている消失訂正においては以下のような演算でエラーの大きさを算出できる。

【００５１】

例えばユーザシンボル中のエラーが、後ろからｉのポジションとｊのポジションで生じているとすると、エラーの大きさをe_i、e_jとすると、これとシンボルS_0、S_1との関係は以下の数１５のようになる。

【００５２】

【数１５】

$$e_i + e_j = S_0$$
$$\alpha^i e_i + \alpha^j e_j = S_1$$

【００５３】

上記数１５からe_i、e_jを求めると以下の数１６のようになるが、これは２元連立方程式として解くことができる。

【００５４】

【数１６】

$$e_i = \frac{\alpha^j \cdot S_0 \cdot S_1}{\alpha^i + \alpha^j}$$

$$e_j = \frac{(\alpha^j \cdot S_0 \cdot S_1) \cdot \alpha^{-i}}{1 + \alpha^{j-1}}$$

【００５５】

次に、検査シンボルE_0に誤りe_jがあったとすると、以下の数１７のようになり、エラーの大きさe_iは１次方程式で解くことができる。

【００５６】

【数１７】

$$e_i + e_j = S_0$$
$$\alpha^i e_i = S_1$$

【００５７】

同様にユーザシンボルに３箇所のエラーe_i，e_j，e_kがあったときには、数１８に示す３元連立方程式を解くことによりエラーの大きさe_i（数１９）を求めることができる。

【００５８】

【数１８】

$$e_i + e_j + e_k = S_0$$
$$\alpha^i e_i + \alpha^j e_j = S_1$$
$$\alpha^{2i} e_i + \alpha^{2j} e_j = S_2$$

【００５９】

【数１９】

$$e_i = \frac{\alpha^j \cdot \alpha^k \cdot S_0 \cdot \alpha^j \cdot S_1 \cdot \alpha^k \cdot S_1 + S_2}{(-\alpha^i + \alpha^j)(-\alpha^i + \alpha^k)}$$

$$e_j = \frac{(\alpha^j \alpha^k S_0 \alpha^j S_1 \alpha^k S_1 + S_2)\alpha^{-j-k}}{(1-\alpha^{i-j})(1-a^{i-k})}$$

【００６０】

同様にしてe_j，e_kを演算できる。

上記の演算において、原始多項式の次数が高くなるにしたがって逆元の計算が膨大なものになる。よって予め決まった値以内になるように変形しておいてテーブルルックアップ方式で逆元を参照することが速度的に有利である。この逆元に関しては、次の２種類のテーブル（参照部）を持てば十分である。

【００６１】

a）α^1、α^2、…、α^k（kはユーザーシンボル数）

b）$1+\alpha^1$、$1+\alpha^2$、…、$1+\alpha^k$（kはユーザーシンボル数）

【００６２】

【発明の効果】

以上のように本発明では、ガロア体の原始多項式を用いたエラー訂正用の演算を高速で行えるように成る。

【図面の簡単な説明】

【図１】

α乗算回路の一例を示すブロック図

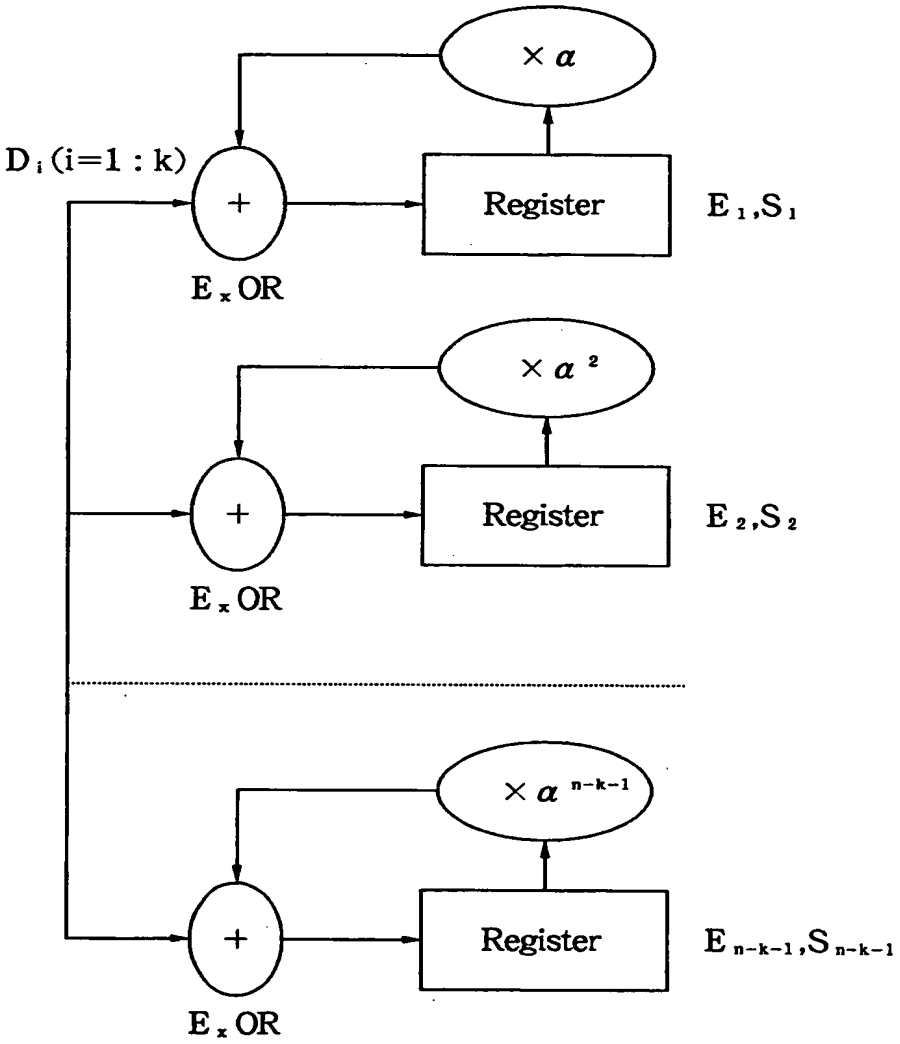
【図２】

α乗算回路の一例を示すブロック図

【書類名】　　図面

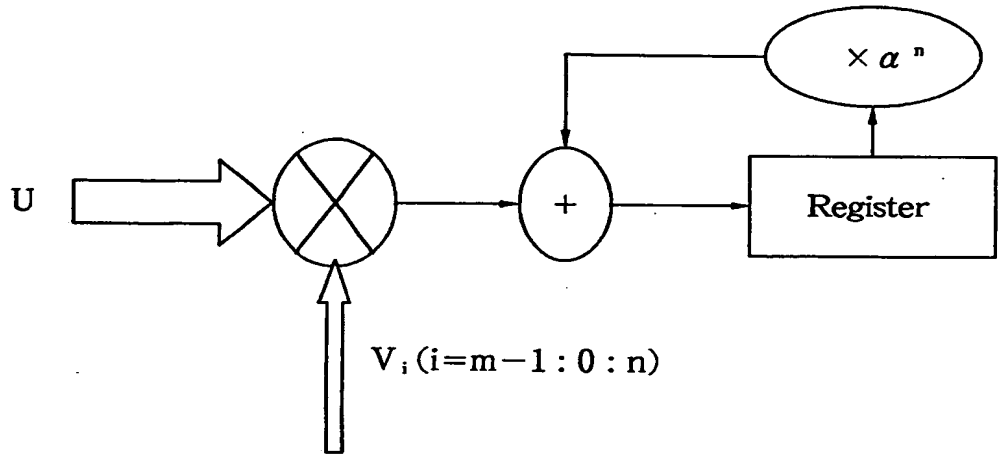
【図１】

図１


×α
Di(i=1：k)
+
Register
E1,S1
ExOR
×α2
+
Register
E2,S2
ExOR
×αn-k-1
+
Register
En-k-1,Sn-k-1
ExOR

【図２】

図２



【書類名】　要約書

【要約】

【課題】　記録媒体にデータをエンコードしまたデコードする場合のシンボルの演算は、ガロア体の原理多項式の元であるαをべき乗算しなくてはならないため、１６ビットや３２ビットのアクセス幅の場合、参照すべきテーブルの容量が大きくなりすぎる。

【解決手段】　前記αにおいて、α^{i}の演算を行うため、ｉビットのシフト演算部と2^{i}の数の参照テーブルを設けた。これにより、小容量のメモリで参照テーブルを実現でき、高速演算が可能となる。

【選択図】　なし

出　願　人　履　歴　情　報

識別番号　　〔０００００１００９８〕

１．変更年月日　　１９９０年　８月２７日
［変更理由］　新規登録
住　所　東京都大田区雪谷大塚町１番７号
氏　名　アルプス電気株式会社